

Vector Prism: Animating Vector Graphics by Stratifying Semantic Structure

686

Supplementary Material

687

Contents

688

A	Extended Related Work	2	689
B	Confidence Bounds for Reliability Estimation	2	690
	B.1 Decomposing Agreements and Rank-One Structure	2	691
	B.2 Reliability Estimation via Eigenvector Analysis	3	692
	B.3 Bayes Rule vs. Majority Voting: Error Bounds	3	693
C	GPT-Human Alignment on Video–Text Alignment	6	694
D	Dataset Composition and Coverage	7	695
E	VLM Prompts for Planning and Animation Generation	8	696
F	Restructuring SVG Files with Semantic Labels	10	697

Please find the `index.html` in the supplementary material if you faced problems loading the video in the main paper.

698

A. Extended Related Work

A.1. Vector graphics generation and decomposition

Beyond animation, the broader field of vector graphics generation has mainly focused on vectorizing raster images. DeepSVG [3] introduced a hierarchical generative network that jointly models both the structure and appearance of vector graphics, enabling controllable generation through learned latent representations. More recently, LayerPeeler [34] proposed an autoregressive approach to decompose raster images into layered vector representations, demonstrating that careful layer-wise decomposition can produce more interpretable and editable vector graphics. Yuan *et al.* [41] extended these ideas to animated stickers, introducing a resource-efficient dual-mask training framework that generates multi-frame animations while maintaining computational efficiency. While our method assumes a user-provided SVG as input, it can be seamlessly combined with image-to-SVG or text-to-SVG models to synthesize SVG animations directly from images or text, eliminating the need for manually authored SVGs.

A.2. Domain-specific SVG understanding

The challenge of understanding structured visual representations extends to specialized domains such as data visualization, such as graphs and charts. Chen *et al.* [5] developed Mystique, a system for deconstructing SVG charts to enable layout reuse, demonstrating that reverse-engineering the semantic structure of charts requires domain-specific parsing strategies. Building on this, VisAnatomy [6] provided a large-scale corpus of SVG charts with fine-grained semantic labels, establishing benchmarks for chart understanding tasks. These works highlight that even within the SVG domain, different application areas (charts vs. illustrations vs. icons) require tailored approaches to semantic understanding. Our framework focuses on general-purpose illustrations and icons, where semantic parts correspond to visual objects rather than data encodings.

A.3. Language models for design tasks

Recent work has explored leveraging large language models for various design tasks beyond vector graphics. Layout-Prompter [14] demonstrated that LLMs can be awakened to perform layout design through carefully crafted prompts that encode spatial relationships and design principles. PosterO [12] extended this to content-aware layout generation by structuring layout trees in a way that enables language models to reason about hierarchical spatial arrangements. These works share our core insight that restructuring visual representations to align with how language models process information is crucial for enabling reliable generation. Taken together, these insights suggest that LLMs are not inherently incapable of design [38], and rather, their potential emerges when design representations are aligned with the natural language structures they are trained to process.

B. Confidence Bounds for Reliability Estimation

In this section, we provide the formal justification for the statistical inference framework described in Section 3.3. We first show how to derive the underlying reliabilities from pairwise agreement (B.1, B.2) and then prove that using these reliabilities in a Bayes-weighted vote is provably superior to a standard majority vote (B.3).

B.1. Decomposing agreements and deriving the rank-one structure

The goal here is to formalize the relationship between the observable quantity (the agreement patterns A_{ij}) and the hidden quantity we care about (the individual reliability of each method, p_i).

Lemma B.1 (Agreement). *Under the symmetric Dawid-Skene model, $A_{ij} = p_i p_j + \frac{(1-p_i)(1-p_j)}{k-1} = \frac{1}{k} + \frac{k}{k-1} \delta_i \delta_j$ for $i \neq j$, with $\delta_i = p_i - \frac{1}{k}$.*

Proof. The probability of agreement $A_{ij} = \Pr[s_i = s_j]$ is the sum of two mutually exclusive cases. One case where both methods are correct ($p_i p_j$), and another case where both methods are incorrect but agree on the same wrong label, which sums to $\frac{(1-p_i)(1-p_j)}{k-1}$. The second expression is obtained by substituting $p_i = \frac{1}{k} + \delta_i$ and $1 - p_i = \frac{k-1}{k} - \delta_i$ into the first expression and simplifying the resulting algebra. ■

Proposition 1 (Rank one). *Let $B_{ij} = A_{ij} - \frac{1}{k}$ ($i \neq j$) and $B_{ii} = 0$. Then $\mathbb{E}[B] = \frac{k}{k-1} \delta \delta^\top$ on off-diagonals (rank one).*

Proof. By definition, the centered agreement matrix entry is $B_{ij} = A_{ij} - \frac{1}{k}$ for $i \neq j$. Substituting the result from Lemma S.1 for A_{ij} :

$$B_{ij} = \left(\frac{1}{k} + \frac{k}{k-1} \delta_i \delta_j \right) - \frac{1}{k} = \frac{k}{k-1} \delta_i \delta_j$$

This is the (i, j) -th entry of the matrix $\frac{k}{k-1} \delta \delta^\top$. Since $\delta \delta^\top$ is the outer product of the vector δ with itself, its rank is one, assuming δ is non-zero. ■

B.2. Reliability estimation via eigenvector analysis

This is the “recovery” part of our proof. Now that we have established a link between agreement and skill (B and δ), we need to show we can actually solve for δ . Theorem S.3 confirms that we can exploit this rank-one structure to reliably estimate the skill vector δ from our empirical data \hat{B} using standard linear algebra techniques, which is finding the top eigenvector.

Theorem B.2 (Quality Guarantee for Estimated Skill). *Let $\hat{B} \in \mathbb{R}^{M \times M}$ be the empirical centered agreement matrix built from n cases and M rendering methods. If each pairwise agreement is estimated within $\pm \varepsilon$, then with probability $\geq 1 - \eta$,*

$$\|\hat{\delta} - c \delta\|_2 \leq C \left(\sqrt{\frac{M}{n}} + \varepsilon \right),$$

for some confidence bound η , scale $c > 0$, and constant C depending only on k . Furthermore, $c = \sqrt{\lambda_1(k-1)/k}$ where λ_1 is the top eigenvalue of \hat{B} .*

Proof. By definition, the centered agreement matrix entry is $B_{ij} = A_{ij} - \frac{1}{k}$ for $i \neq j$. Substituting the result from Lemma S.1 for A_{ij} :

$$B_{ij} = \left(\frac{1}{k} + \frac{k}{k-1} \delta_i \delta_j \right) - \frac{1}{k} = \frac{k}{k-1} \delta_i \delta_j$$

This means the matrix $\mathbb{E}[B]$ is a constant factor $\frac{k}{k-1}$ times the matrix $\delta \delta^\top$. The outer product of any vector with itself, $\delta \delta^\top$, is mathematically a rank-one matrix. This rank-one property is critical because it means the vector δ (our reliability or ‘skill’ vector) must be proportional to the dominant eigenvector of B , enabling its recovery in Theorem S.3. ■

B.3. Bayes decision rule and error bounds vs. Majority voting

This section proves that weighting VLM responses by their inferred reliability is **statistically superior** to simple majority voting. Corollary B.5 shows that the Bayes-weighted method achieves a strictly better error exponent whenever VLM reliabilities differ.

B.3.1. Setup and the log-likelihood ratio

Fix the true label $y^* \in \{1, 2, \dots, k\}$ and any competitor label $y \neq y^*$. For each method i , recall that p_i is the probability of correct classification. Define:

- $q_i \triangleq \frac{1-p_i}{k-1}$ (probability of any specific wrong label)
- $d_i \triangleq p_i - q_i = \frac{k p_i - 1}{k-1}$ (discrimination parameter)
- $w_i = \log \frac{p_i}{q_i} = \log \frac{(k-1)p_i}{1-p_i}$ (Bayes weight, or log-likelihood-ratio)

For each observation s_i (method i ’s output), define the log-likelihood ratio:

$$Z_i = w_i \cdot \mathbf{1}[s_i = y^*] - w_i \cdot \mathbf{1}[s_i = y] = \begin{cases} +w_i & \text{if } s_i = y^* \\ -w_i & \text{if } s_i = y \\ 0 & \text{otherwise} \end{cases}$$

The Bayes decision rule prefers y^* over y when $\sum_i Z_i > 0$. An error occurs when $\sum_i Z_i \leq 0$ despite y^* being true.

Lemma B.3 (Properties of Z_i). *Under the true label y^* : (1) $Z_i \in [-w_i, w_i]$, (2) $\mathbb{E}[Z_i \mid y^*] = w_i d_i$, and (3) the Z_i are independent across methods.*

Proof. Boundedness is immediate from the definition. For the expected value, given y^* , method i outputs y^* with probability p_i (giving $Z_i = w_i$) and outputs y with probability q_i (giving $Z_i = -w_i$). Thus:

$$\mathbb{E}[Z_i \mid y^*] = p_i \cdot w_i + q_i \cdot (-w_i) = w_i(p_i - q_i) = w_i d_i.$$

Independence follows from the conditional independence assumption in the Dawid-Skene model. ■

*The underlying mathematics, based on the Davis-Kahan theorem, provides a strong upper limit on the error between our calculated skill vector ($\hat{\delta}$) and the true skill vector (δ). This error is confirmed to decrease as we process more SVG primitives (n).

774 **B.3.2. Error bound for Bayes decision rule**

Theorem B.4 (Hoeffding bound for Bayes LLR).

$$\Pr \left[\sum_{i=1}^m Z_i \leq 0 \mid y^* \right] \leq \exp \left(- \frac{(\sum_{i=1}^m w_i d_i)^2}{2 \sum_{i=1}^m w_i^2} \right).$$

Proof. We apply Hoeffding's inequality for bounded independent random variables. Since $Z_i \in [-w_i, w_i]$ with range $(b_i - a_i) = 2w_i$, Hoeffding's inequality gives:

$$\Pr \left[\sum_{i=1}^m (Z_i - \mathbb{E}[Z_i]) \leq -t \right] \leq \exp \left(- \frac{2t^2}{\sum_{i=1}^m 4w_i^2} \right).$$

The error event $\sum Z_i \leq 0$ can be rewritten as $\sum (Z_i - \mathbb{E}[Z_i]) \leq -\sum \mathbb{E}[Z_i]$. Setting $t = \sum_{i=1}^m w_i d_i = \sum_{i=1}^m \mathbb{E}[Z_i]$ and substituting:

$$\Pr \left[\sum_{i=1}^m Z_i \leq 0 \right] \leq \exp \left(- \frac{2(\sum_{i=1}^m w_i d_i)^2}{4 \sum_{i=1}^m w_i^2} \right) = \exp \left(- \frac{(\sum_{i=1}^m w_i d_i)^2}{2 \sum_{i=1}^m w_i^2} \right).$$

782

The exponent $\frac{(\sum w_i d_i)^2}{2 \sum w_i^2}$ quantifies how fast the error probability decays. Larger exponents mean exponentially smaller error rates.

785 **B.3.3. Comparison with majority voting and proof of superiority**

Majority voting uses uniform weights $w_i^{\text{MV}} \equiv 1$, yielding error exponent $\frac{(\sum d_i)^2}{2m}$. We now show that Bayes weighting is strictly better when reliabilities differ.

Theorem B.5 (Improvement over majority voting). *In the small-error regime where $|p_i - \frac{1}{k}| \ll 1$, the approximations $w_i \approx \frac{k^2}{k-1} \delta_i$ and $d_i \approx \frac{k}{k-1} \delta_i$ (where $\delta_i = p_i - \frac{1}{k}$) imply $w_i \approx k d_i$. The Bayes error exponent then satisfies:*

$$\text{Exponent}_{\text{BV}} \approx \frac{1}{2} \sum_{i=1}^m d_i^2 = \frac{m}{2} [(\text{Mean}(d))^2 + \text{Var}(d)] \geq \text{Exponent}_{\text{MV}} = \frac{(\sum d_i)^2}{2m},$$

with equality if and only if all d_i are equal. The improvement factor is

$$\frac{\text{Exponent}_{\text{BV}}}{\text{Exponent}_{\text{MV}}} \approx 1 + \frac{\text{Var}(d)}{(\text{Mean}(d))^2},$$

quantifying the benefit of exploiting heterogeneity in method reliabilities.

Proof. First, we derive the weight approximation. For $p_i = \frac{1}{k} + \delta_i$ with small δ_i :

$$w_i = \log \frac{(k-1)p_i}{1-p_i} = \log \frac{(k-1)(\frac{1}{k} + \delta_i)}{\frac{k-1}{k} - \delta_i} = \log \frac{1 + k\delta_i}{1 - \frac{k\delta_i}{k-1}}.$$

Using Taylor expansions $\log(1+x) \approx x$ and $(1-y)^{-1} \approx 1+y$:

$$w_i \approx k\delta_i + \frac{k\delta_i}{k-1} = \frac{k^2\delta_i}{k-1}.$$

Similarly, $d_i = \frac{kp_i-1}{k-1} = \frac{k\delta_i}{k-1}$, so $w_i \approx k d_i$. Now compute the Bayes exponent using $w_i \approx k d_i$:

$$\text{Exponent}_{\text{BV}} = \frac{(\sum w_i d_i)^2}{2 \sum w_i^2} \approx \frac{(k \sum d_i^2)^2}{2k^2 \sum d_i^2} = \frac{\sum d_i^2}{2}.$$

Using the variance decomposition $\sum d_i^2 = m(\text{Mean}(d))^2 + m \cdot \text{Var}(d)$:

$$\text{Exponent}_{\text{BV}} \approx \frac{m}{2} [(\text{Mean}(d))^2 + \text{Var}(d)].$$

For majority voting with $w_i = 1$:

$$\text{Exponent}_{\text{MV}} = \frac{(\sum d_i)^2}{2m} = \frac{m^2(\text{Mean}(d))^2}{2m} = \frac{m(\text{Mean}(d))^2}{2}.$$

The difference is $\text{Exponent}_{\text{BV}} - \text{Exponent}_{\text{MV}} \approx \frac{m \cdot \text{Var}(d)}{2} \geq 0$, with equality only when $\text{Var}(d) = 0$ (all d_i equal). The improvement factor is:

$$\frac{\text{Exponent}_{\text{BV}}}{\text{Exponent}_{\text{MV}}} = \frac{\frac{m}{2} [(\text{Mean}(d))^2 + \text{Var}(d)]}{\frac{m(\text{Mean}(d))^2}{2}} = 1 + \frac{\text{Var}(d)}{(\text{Mean}(d))^2}. \quad \blacksquare$$

Remark B.6. The improvement factor $1 + \frac{\text{Var}(d)}{(\text{Mean}(d))^2}$ shows that Bayes weighting provides the most benefit when method reliabilities are heterogeneous. If all methods have identical reliability, both approaches are equivalent. The more diverse the reliabilities, the greater the advantage of properly weighting methods by their estimated skill.

C. GPT-Human Alignment on Video-Text Alignment

We assess the alignment between our user study and the GPT-T2V metric in order to validate the reliability of the GPT based evaluation. In particular, we compare pairwise preferences and measure how often the metric selects the same animation as human participants. We find that GPT’s preferences (*i.e.*, cases where GPT assigns a higher score to one animation than the other) agree with user preferences in 83.4% of the pairs, which indicates a strong correspondence between the automatic and human judgments. In comparison, the CLIP-T2V metric, which operates without any external API services, reaches only 53.4% agreement with the user study responses. This substantial gap suggests that GPT-T2V captures human perceptual preferences much more faithfully and therefore provides a more reliable proxy for human evaluation in our setting.

We observe that state-of-the-art LLMs demonstrate a robust ability to interpret simple animations and reason about their motion. When guided by clear evaluation criteria, such as those provided in Figure 9, these models exhibit a high degree of alignment with human judgments. Although GPT-5 is also the model used to generate our animations, its role as an evaluator is fundamentally different. In practice, using LLMs as judges is often more straightforward and reliable than using them as generators, as evaluation requires consistency and comparative reasoning rather than creative synthesis.

You are evaluating whether a video matches a given text description.

Text Description: "{text_prompt}"

Here are frames sampled from the video. Evaluate how well it depicts the given description.

Where the score represents:

- 90-100: Perfect match, video clearly depicts the description
- 70-89: Good match, most elements are present
- 50-69: Partial match, some elements are present
- 30-49: Weak match, few elements present
- 0-29: No match, video does not depict the description.

Provide your response in EXACTLY this json format:

```
```json
{
 "score": score_value,
 "reasoning": "brief explanation"
}
```

**GPT-T2V**

Figure 9. Prompt templated used for GPT-T2V evaluation.

D. Dataset Composition and Coverage

Our test dataset comprises 114 hand-crafted animation instructions across 57 unique SVG files, with each SVG file receiving an average of two distinct animation scenarios. These examples were meticulously designed to reflect the diverse animation needs encountered in modern web development. As shown in Table 2, our dataset spans six thematic categories, with particularly strong representation in Nature/Environment (31.6%) and Objects/Miscellaneous (26.3%), ensuring broad coverage of visual content types commonly found in web interfaces. From tech logos and brand animations to natural phenomena and user interface elements, our dataset encompasses the full spectrum of SVG animation use cases. Furthermore, Table 3 demonstrates our intentional focus on varied interaction patterns, with Appearance/Reveal animations (28.1%) and State Transition effects (13.2%) representing critical components of modern web user experiences. The substantial presence of Organic/Natural Movement (12.3%) and Rotational Movement (8.8%) patterns reflects our commitment to including both subtle, life-like animations and dynamic, attention-grabbing effects. This careful curation ensures that our test dataset not only provides comprehensive coverage but also accurately represents the practical animation requirements of contemporary web applications, from loading indicators and state feedback to decorative enhancements and interactive storytelling.

Table 2. Distribution of Subject Themes in Test Dataset

Subject Theme	Count	%
Nature/Environment	36	31.6
Objects/Miscellaneous	30	26.3
UI/Interface Elements	18	15.8
Tech Logos/Brands	12	10.5
Animals/Characters	10	8.8
Faces/Emojis	8	7.0
Total	114	100.0

Table 3. Distribution of Interaction Patterns in Test Dataset

Interaction Pattern	Count	%
Other/Mixed	43	37.7
Appearance/Reveal	32	28.1
State Transition	15	13.2
Organic/Natural Movement	14	12.3
Rotational Movement	10	8.8
Total	114	100.0

## E. VLM Prompts for Planning and Animation Generation

Our animation pipeline relies on two complementary VLM prompts, one for planning and one for per-class animation generation.

The model is instructed to avoid generic SVG terms and to instead use intuitive, role-based identifiers, and to write a short, human-interpretable description of the intended motion of each part. This prompt focuses entirely on *semantic intent* and it does not require the model to understand SVG syntax, only to reason visually and symbolically about what should happen.

The second prompt is invoked once for each semantic class produced during restructuring. It receives three ingredients, the restructured SVG, all previously generated CSS (so it can remain consistent), and the animation plan for that particular class. Its role is purely *syntactic* and translate one actor's high-level plan into concrete, production-safe CSS. To avoid conflicts across iterative generations, the prompt enforces a strict "lanes" convention in which each motion component (translation, rotation, scale, opacity, blur, etc.) is expressed through typed CSS custom properties rather than direct transform declarations in keyframes. A single composer rule per class then assembles these properties into the final transform. This ensures that new animations never overwrite existing ones, allowing independent motions to compose reliably across multiple generation passes.

The two prompts divide responsibilities cleanly, the planner performs semantic reasoning, and the per-class generator performs structured code synthesis. This separation avoids the common failure modes where a single prompt must juggle visual interpretation, HTML/SVG structure, and CSS constraints simultaneously. The lanes system further guarantees that iterative code generation remains stable, that different motions do not collide, and that long CSS files can be produced incrementally without exceeding model context limits.

This is an SVG image.

Your task is to generate animation plans for the individual elements in the image based on the following instruction:  
{instruction}

The goal is to create a **high-quality, smooth, and visually engaging SVG animation** suitable for web display.

Please follow these guidelines:

- **Animate elements individually or in thoughtfully grouped sets**. Each group should share similar motion or timing.
- **Avoid awkward or robotic movement** unless intentional. The animation should feel natural and dynamic.
- An element can be animated through changes in attributes like position, scale, rotation, color, opacity, path, etc.
- In case elements of similar type (e.g. trees, stars, clouds) is expected to have different animations, treat them as **distinct elements** (e.g., 'left\_tree', 'foreground\_star', 'background\_cloud'), but if they should share an animation, treat them as a single element.
- Keep the number of elements in a manageable range (e.g., 5-10) so that the animation is not overly complex.
- Avoid using generic or SVG/HTML tag-based names (e.g., 'circle', 'rect', 'path', 'body'). Instead, use meaningful identifiers based on position or visual role. Also do not use special characters that could interfere with JSON formatting or directory paths (e.g., '#', '.', '/', '\\', ':').
- If an element has no animation, it should be explicitly stated as such.
- Please do not plan accessibility features or interactivity.
- Do not include any runtime-only classes (e.g., .impact, .flight, .play) in your plan. The animation must animate immediately on page load with no manual steps or JS triggers.

Please respond in the following JSON format:

```

{
 "json":
 {
 "element_name_1": Animation plan for element_name_1,
 "element_name_2": Animation plan for element_name_2,
 ...
 "element_name_n": Animation plan for element_name_n,
 }
}

```

## Animation Planner

Figure 10. Prompt template used for planning animations. The output is a JSON formatted dictionary of semantic categories and their animation plan.



You are a CSS animation expert tasked with creating animations for SVG elements.  
The first image is the entire SVG file.

We have animated the following elements in the SVG:

```
```html
{previous_html}
```
```

Now, we are currently focusing on animating the `{class\_name}` class within the SVG, which is rendered in the second image.

Animation plan for `{class\_name}` is as follows:

```
{animation_plan}
```

Please generate CSS animation code for the SVG element with class '{class\_name}'.

Requirements:

- Create keyframe animations that are timed and executed harmoniously with existing animations in the SVG.
- Animation should be smooth, optimized, and appropriate for web performance.
- Style should be elegant and subtle unless dramatic effects are specifically requested.
- Avoid naming conflicts with existing keyframes or animation properties.
- Include compact comments regarding coherence with other animations where relevant.
- Coordinates and transform origins must be derived based on the actual layout of the entire SVG. Account for the spatial relationship between {class\_name} and other animated elements to avoid visual collisions, clipping, or misalignment. Use relative positions where appropriate.
- Refrain from modifying or duplicating any existing CSS code.
- Be mindful of the performance implications of your animations, especially for complex SVGs with multiple animated elements.
- Make all animations self-contained. Do NOT gate keyframes behind runtime-only classes (e.g., .impact, .flight, .play). The delivered file must animate immediately on page load with no manual steps.

Collision avoidance considerations:

- Never write 'transform' inside @keyframes. Write Custom properties only.
- Use the lanes pattern with these naming convention: --{class\_name}-tx1/tx2, --{class\_name}-ty1/ty2, --{class\_name}-rot1/rot2, --{class\_name}-sx1/sx2, --{class\_name}-sy1/sy2, --{class\_name}-op1/op2, --{class\_name}-blur1/blur2, --{class\_name}-stroke1/stroke2, --{class\_name}-bright1/bright2.
- If these @property declarations or the .{class\_name} composer rule are missing, add them ONCE.
- Put new motion on the next free lane(s). Do NOT edit existing lanes.
- Use animation-\* longhand. If multiple animations, provide comma-separated lists with aligned indexes.

Please respond in the following format:

```
```html
<style>
  /* CSS code goes here */
</style>
```
```

## Animation Generator

Figure 11. Prompt template used for generating animations. CSS codes are generated in a cascaded manner to bypass generation token length limits.

## 855 F. Restructuring SVG Files with Semantic Labels

856 Once we obtain semantic labels for all primitives, we reorganize the SVG structure by regrouping primitives according to  
 857 their labels wherever possible. This is non-trivial because existing SVG groupings are tightly coupled to the rendering order,  
 858 and naively introducing new groups can disrupt this order and alter the final appearance.

859 Nevertheless, restructuring is crucial for enabling meaningful motion, as primitives that belong to the same semantic  
 860 group can share attributes such as rotation axes, timing, and other animation parameters. To safely regroup primitives, we  
 861 first flatten the SVG structure and ungroup all nested groups, while transferring group properties to the child primitives, so  
 862 that the rendering appears identical to the original. Then, we estimate the spatial extent (area) occupied by each primitive  
 863 and use this to detect conflicting merges. Next, we merge primitives with the same semantic label only when doing so  
 864 introduces no conflicts with any primitives in between them in the rendering order. Finally, we augment each resulting group  
 865 with metadata, including its bounding box, geometric center, and parent-child relationships, which we later use to drive  
 866 animation. We describe the steps in Algorithm 1 and plan to make all the implementation fully public upon acceptance.

---

**Algorithm 2** Pseudocode for the SVG file restructuring process using the predicted semantic labels.

---

- 1: **Inputs** SVG  $S$ , predicted label  $\hat{y}(x)$  for each primitive  $x$
  - 2: **Output** regrouped SVG  $S'$
  - 3: **Flatten**
  - 4: Traverse  $S$  in original paint order and build a list  $E = \{(e, \text{idx}, \ell, B)\}$   
 $e$  is a cloned primitive with inherited properties baked in  
 $\text{idx}$  is the original paint index  
 $\ell = \hat{y}(e)$  is appended as the final class token  
 $B$  is a screen-space bounding box
  - 5: **Regroup by label with a barrier test**
  - 6: **for** each label  $\ell$  **do**
  - 7:    $I_\ell \leftarrow$  indices of  $E$  with label  $\ell$  in ascending paint order
  - 8:   Greedily form groups  $G[\ell]$  over  $I_\ell$  using the rule:  
     a candidate  $j$  can join current group  $G$  if no element of a different label  
     whose index lies between  $\min(G \cup \{j\})$  and  $\max(G \cup \{j\})$   
     overlaps any member of  $G \cup \{j\}$  in screen space
  - 9: **end for**
  - 10: **Compose regrouped SVG**
  - 11: Create  $S'$  with original attributes and non-drawables copied verbatim
  - 12: For each group  $g$  in order of its earliest index:  
     emit a  $\langle g \rangle$  with class  $\ell$ -group or  $\ell$ -group- $k$   
     append members in original relative order  
     write light metadata: bounds, geometric center, paint-order index  
     optionally add parent and children links from the plan
  - 13: **return**  $S'$
-

## References

- [1] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023. 2
- [2] Mu Cai, Zeyi Huang, Yuheng Li, Utkarsh Ojha, Haohan Wang, and Yong Jae Lee. Leveraging large language models for scalable vector graphics-driven image understanding. *arXiv preprint arXiv:2306.06094*, 2023. 2
- [3] Alexandre Carlier, Martin Danelljan, Alexandre Alahi, and Radu Timofte. Deepsvg: A hierarchical generative network for vector graphics animation. *NeurIPS*, 2020. 2
- [4] Sumit Chaturvedi, Michal Lukáč, and Siddhartha Chaudhuri. Regroup: Recursive neural networks for hierarchical grouping of vector graphic primitives. *arXiv preprint arXiv:2111.11759*, 2021. 2
- [5] Chen Chen, Bongshin Lee, Yunhai Wang, Yunjeong Chang, and Zhicheng Liu. Mystique: Deconstructing svg charts for layout reuse. *IEEE TVCG*, 2023. 2
- [6] Chen Chen, Hannah K Bako, Peihong Yu, John Hooker, Jeffrey Joyal, Simon C Wang, Samuel Kim, Jessica Wu, Aoxue Ding, Lara Sandeep, et al. Visanatomy: An svg chart corpus with fine-grained semantic labels. *arXiv preprint arXiv:2410.12268*, 2024. 2
- [7] Mark Chen. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. 1
- [8] David L Davies and Donald W Bouldin. A cluster separation measure. *IEEE TPAMI*, 2009. 8
- [9] Alexander Philip Dawid and Allan M Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 1979. 2, 4
- [10] Alexander Grah. The animate package, 2011. 6
- [11] Vision Cortex Group. Vtracer, 2020. 8
- [12] HsiaoYuan Hsu and Yuxin Peng. Postero: Structuring layout trees to enable language models in generalized content-aware layout generation. In *CVPR*, 2025. 2
- [13] Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. Competition-level code generation with alphacode. *Science*, 378(6624): 1092–1097, 2022. 4
- [14] Jiawei Lin, Jiaqi Guo, Shizhao Sun, Zijiang Yang, Jian-Guang Lou, and Dongmei Zhang. Layoutprompter: Awaken the design ability of large language models. *NeurIPS*, 2023. 2
- [15] Kevin Qinghong Lin, Yuhao Zheng, Hangyu Ran, Dantong Zhu, Dongxing Mao, Linjie Li, Philip Torr, and Alex Jinpeng Wang. Vcode: a multimodal coding benchmark with svg as symbolic visual representation. *arXiv preprint arXiv:2511.02778*, 2025. 3
- [16] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *NeurIPS*, 36, 2023. 1
- [17] Zichen Liu, Yihao Meng, Hao Ouyang, Yue Yu, Bolin Zhao, Daniel Cohen-Or, and Huamin Qu. Dynamic typography: Bringing text to life via video diffusion prior. In *ICCV*, 2025. 2
- [18] Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. Codegen: An open large language model for code with multi-turn program synthesis. In *ICLR*, 2023. 4
- [19] Kunato Nishina and Yusuke Matsui. Sgeditbench v2: A benchmark for instruction-based svg editing. *arXiv preprint arXiv:2502.19453*, 2025. 2, 3
- [20] OpenAI. Gpt-5, 2025. 1, 5
- [21] OpenAI. Sora2 system card, 2025. 5
- [22] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *ICLR*, 2023. 2, 5
- [23] Zeju Qiu, Weiyang Liu, Haiwen Feng, Zhen Liu, Tim Z Xiao, Katherine M Collins, Joshua B Tenenbaum, Adrian Weller, Michael J Black, and Bernhard Schölkopf. Can large language models understand symbolic graphics programs? *ICLR*, 2025. 3
- [24] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*. Pmlr, 2021. 5
- [25] Juan A Rodriguez, Shubham Agarwal, Issam H Laradji, Pau Rodriguez, David Vazquez, Christopher Pal, and Marco Pedersoli. Starvector: Generating scalable vector graphics code from images. In *CVPR*, 2025. 2, 8
- [26] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 2
- [27] Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Hervé Jégou, Patrick Labatut, and Piotr Bojanowski. DINOv3, 2025. 8
- [28] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwei Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wenten Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025. 5
- [29] Haomin Wang, Jinhui Yin, Qi Wei, Wenguang Zeng, Lixin Gu, Shenglong Ye, Zhangwei Gao, Yaohui Wang, Yanting

- Zhang, Yuanqi Li, et al. Internsvg: Towards unified svg tasks with multimodal large language models. *arXiv preprint arXiv:2510.11341*, 2025. 2, 5
- [30] Yi Wang, Yinan He, Yizhuo Li, Kunchang Li, Jiashuo Yu, Xin Ma, Xinhao Li, Guo Chen, Xinyuan Chen, Yaohui Wang, et al. Internvid: A large-scale video-text dataset for multimodal understanding and generation. In *ICLR*, 2024. 5
- [31] Zhenyu Wang, Jianxi Huang, Zhida Sun, Yuanhao Gong, Daniel Cohen-Or, and Min Lu. Layered image vectorization via semantic simplification. In *CVPR*, 2025. 2
- [32] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *NeurIPS*, 2022. 1
- [33] Haoning Wu, Erli Zhang, Liang Liao, Chaofeng Chen, Jingwen Hou, Annan Wang, Wenxiu Sun, Qiong Yan, and Weisi Lin. Exploring video quality assessment on user generated contents from aesthetic and technical perspectives. In *ICCV*, 2023. 5
- [34] Ronghuan Wu, Wanchao Su, and Jing Liao. Layerpeeler: Autoregressive peeling for layer-wise image vectorization. *SIGGRAPH Asia*, 2025. 2
- [35] Ronghuan Wu, Wanchao Su, Kede Ma, and Jing Liao. Aniclipart: Clipart animation with text-to-video priors. *IJCV*, 2025. 2, 5
- [36] Ximing Xing, Juncheng Hu, Guotao Liang, Jing Zhang, Dong Xu, and Qian Yu. Empowering llms to understand and generate complex vector graphics. In *CVPR*, 2025. 2
- [37] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. 1
- [38] Jialin Yang, Dongfu Jiang, Lipeng He, Sherman Siu, Yuxuan Zhang, Disen Liao, Zhuofeng Li, Huaye Zeng, Yiming Jia, Haozhe Wang, et al. Structeval: Benchmarking llms' capabilities to generate structural outputs. *arXiv preprint arXiv:2505.20139*, 2025. 2
- [39] John Yang, Carlos E Jimenez, Alex L Zhang, Kilian Lieret, Joyce Yang, Xindi Wu, Ori Press, Niklas Muennighoff, Gabriel Synnaeve, Karthik R Narasimhan, et al. Swe-bench multimodal: Do ai systems generalize to visual software domains? In *ICLR*, 2025. 1
- [40] Yiyang Yang, Wei Cheng, Sijin Chen, Xianfang Zeng, Fukun Yin, Jiaxu Zhang, Liao Wang, Gang Yu, Xingjun Ma, and Yu-Gang Jiang. Omnisvg: A unified scalable vector graphics generation model. In *NeurIPS*, 2025. 2, 8
- [41] Zhiqiang Yuan, Ting Zhang, Ying Deng, Jiawei Zhang, Yeshuang Zhu, Zexi Jia, Jie Zhou, and Jinchao Zhang. Rdtf: Resource-efficient dual-mask training framework for multi-frame animated sticker generation. *arXiv preprint arXiv:2503.17735*, 2025. 2
- [42] Peiying Zhang, Nanxuan Zhao, and Jing Liao. Text-to-vector generation with neural path representation. *ACM TOG*, pages 1–13, 2024. 2
- [43] Peiying Zhang, Nanxuan Zhao, and Jing Liao. Style customization of text-to-vector generation with image diffusion priors. In *SIGGRAPH*, pages 1–11, 2025. 2
- [44] Tong Zhang, Haoyang Liu, Peiyan Zhang, Yuxuan Cheng, and Haohan Wang. Beyond pixels: Exploring human-readable svg generation for simple images with vision language models. *arXiv preprint arXiv:2311.15543*, 2023. 2, 3
- [45] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhaghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *NeurIPS*, 2023. 5
- [46] Bocheng Zou, Mu Cai, and Jianrui Zhang Yong Jae Lee. Vg-bench: Evaluating large language models on vector graphics understanding and generation. *EMNLP*, 2024. 2, 3